Chapter 8

Extraction de Structures Arborescentes

Résumé — Dans la partie précédente nous avons présenté un algorithme mettant en oeuvre une collaboration entre le *Fast-Marching* et les *Level-Sets* pour la segmentation. Dans ce chapitre, nous souhaitons présenter une application de cette collaboration spécifiquement dédiée aux structures arborescentes du type arbre vasculaire.
Tout d'abord nous montrons comment le *Fast-Marching* permet de fournir une présegmentation rapide et précise pour les structures arborescentes dans la section 8.3.
Nous utilisons ensuite les *Level-Sets* de la même manière que dans la section 5.4 de la partie précédente.
Finalement nous montrons comment le *Fast-Marching*, déjà utilisé pour l'extraction de trajectoires dans la partie I, permet aussi d'extraire plusieurs trajectoires et de remonter à l'information d'arbre ou de squelette d'un objet tubulaire avec embranchements multiples.

Abstract — In the previous part I, we detailed an algorithm using *Fast-Marching* and *Level-Sets* in a collaborative manner for object segmentation. In this chapter, we introduce an application of this collaboration specifically adapted to tree anatomical structures, like vascular or arterial tree. First of all, we demonstrate in section 8.3 the ability to build a fast and accurate pre-segmentation for those tree structures using a dedicated *Fast-Marching* algorithm.
We further apply the *Level-Sets*, as in section 5.4, for converging to a more accurate solution.
Finally, we show in section 8.4 how the *Fast-Marching* ability to extract trajectories, as used in part I, can be extended to the simultaneous extraction of multiple trajectories, and to obtain the underlying tree structure of a tubular anatomical shape with several branches.

8.1 Introduction

In the first part of the thesis, we have implemented several techniques to extract a trajectory inside a tubular structure. We have shown application of this fast minimal path-extraction process to automatic and interactive methods to extract lineic structures in images. In the second part of the thesis, we have combined fast and accurate methods for shape extraction, using the same kind of grey-weighted distance transform algorithms. We have proved the ability of those techniques to extract surfaces, and to emphasize pathologies, in several applications. In the last part of the thesis, we now want to integrate the path and surface extraction algorithms, in order to present an accurate global framework for the segmentation, the visualization, and the quantification, of anatomical objects. In the previous chapter, we have detailed the algorithmic techniques to obtain representations and measures of our anatomical objects, based on extracted primitives of our objects like shapes and skeletons. In this chapter, we will present the basic framework, and extend its possibilities to the detection of tree-like structures, and their corresponding set of multiple trajectories, in order to enhance measures and visualization of pathologies of any tube-shaped object.

This chapter will be illustrated by applications of the algorithms presented on the segmentation and quantification of vessels in contrast-enhanced 3D medical images.

8.2 Motivation

We have seen in part II a method to use front competition for image segmentation. This process involved to visit the whole image domain, and was not tuned for a particular category of objects. Moreover, in huge images, as multi-slice CT scanners (see application to lungs images in chapter 9.2), the visit of the whole image cannot be done in interactive time.

8.2.1 Tree extraction

In this chapter, we are focusing on the extraction of thin tubular structures. Our algorithms can be dedicated to this particular category of tube-shaped objects. If the propagation of a front could be restricted to the part of the image occupied by those structures, the computing time could be divided by almost 5, since vessels in a typical MR-Angiography image do not exceed 10% of the whole volume.

In chapter 2, we have developed an algorithm that can be the basis of this kind of tubular shape extraction object: a technique to evolve a front inside an object of interest and compute at the same time the Euclidean distance to the start point. It was used to reduce the user interaction to locating only one extremity of the path inside a tubular structure. This Euclidean distance can be used to stop the front propagation inside the desired object. If we have precise knowledge of its length, we can decide to stop when this given length has been reached in the expression of the Euclidean path length computation, as explained in section 2.2.3. The result of this technique is shown in figure 8.1.

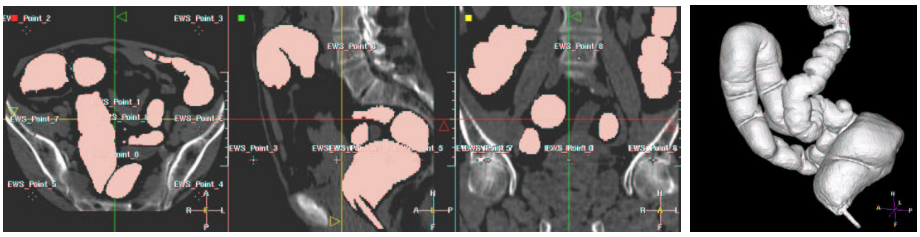

Figure 8.1. Segmenting the colon volume with simple front propagation: as in the virtual endoscopy facility, the user locate a starting point at one particular recognizable part of the colon, then a front is propagated from this seed point until a maximum path length is reached. Left image represents the datasets where the intersection with the segmented object is visible in pink. Right image is the 3D volume rendering of the final segmentation.

However, classical segmentation problems do not provide an excellent contrast like the air-filled colon on a CT scanner, and the propagation cannot stick to the object walls, as it is shown in figure 3.6. For example, if we apply the same kind of propagation in the dataset shown in figure 3.12 for the endoscopy application in chapter 3, the corresponding wave propagation looks like figure 8.2. The front floods

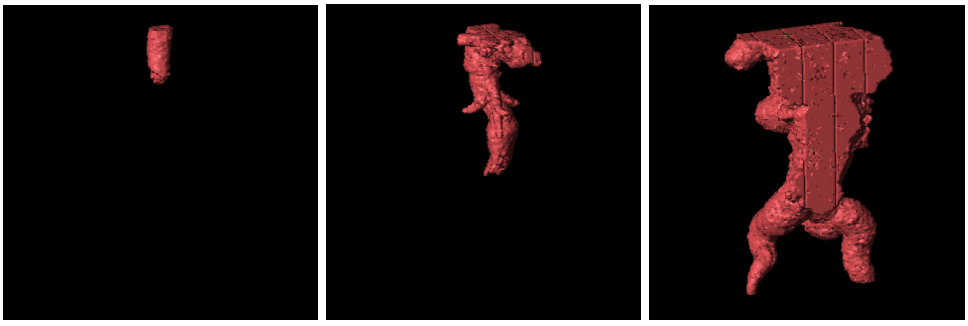

Figure 8.2. Wave propagation inside the aorta MR dataset: These three images represent different steps of the propagation inside the aorta MR dataset using *Eikonal equation* equation with a potential similar to the one defined for the endoscopy application (a simple function of the grey levels either linear or non-linear).

outside the object and cannot be used as an initialization step for a more complex segmentation, like the combination of the *Fast-Marching* and the *Level-Sets* which was presented in the previous part.

In the following section, we will present a new algorithm, based on the *Fast-Marching* and dedicated to a *quick and dirty* segmentation of the tree structures in 3D medical images.

8.3 Design of an adequate initialization algorithm

We have shown the possibility to provide efficiently an initialization for more complicated methods in the previous part of the thesis. Setting up a framework for the visualization and the quantification of thin tubular structures, based on the same combination of the *Fast-Marching* and the *Level-Sets*, we show in this section how the previous initialization step, which is not tuned for this kind of thin and long objects, can be specifically optimized for this target.

8.3.1 Propagation Freezing for Thin Structures

Freezing a voxel during front propagation is to consider that it has reached the boundary of the structure. When the front propagates in a thin structure, there is only a small part of the front, which we could call the "head" of the front, that really moves. Most of the front is located close to the boundary of the structure and moves very slowly. For example voxels that are close to the starting point, the "tail" of the front, are moving very slowly. However, since the structure may be very long, in order for the "head" voxels to reach the end of the structure, the "tail" voxels may flow out of the boundary since their speed is always positive. This is illustrated in the example of figure 3.12. If we apply fast marching in the dataset shown in figure 3.12-top with a potential based on the gray level with contrast enhancement the corresponding wave propagation looks like figure 8.2. The front floods outside the object and does not give a good segmentation.

For these reasons, it is of no use to make some voxels participate in the computation of the arrival time in *Eikonal equation* by setting their speed to zero, which we call *Freezing*. First step is to design the appropriate criterion for selecting voxels of the front which needs *Freezing*.

Concerning the application to the tree tracking, the several improvements brought by this method are

- to accelerate the computations, by visiting a very small number of voxels during propagation;
- to enable the segmentation of thin tubular structures;
- therefore enabling the centering inside those tubular structures.

First step is do design the appropriate criterion for *Freezing* voxels of the front. We illustrate this *Freezing* principle on a synthetic branching structure in 2D.

Synthetic test problem

A synthetic example of a tree structure is shown in figure 8.3. In this case, setting an initial seed point at the hierarchy, we would like to extract in a very fast process the multiple branches of the structures, and its corresponding skeletonization, in a single process. Figure 8.3 shows the result of the classical front propagation technique with the *Fast-Marching* coupled with a maximum Euclidean path length stopping criterion. The action map displayed clearly indicates that the domain visited is a whole

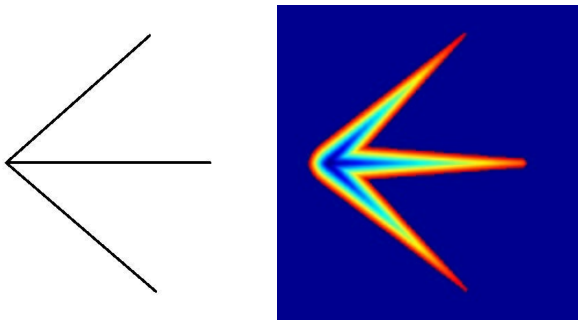

Figure 8.3. Synthetic test problem: The left image is the medium where a front has been propagated, starting at the root of the three branches, and stopping when a maximum distance criterion of 300, computed according to method described in section 2.2.3, has been reached; Right image is the corresponding action map.

"blob-like" structure where the underlying tubular shape is somehow lost. Therefore, tracking a minimal path from the regional maxima of the action map will not lead for sure to paths that stay inside the object of interest. It emphasizes the little use of this method, without a clear constraint on the domain of points visited.

Using Time for Freezing

The heuristic presented in this section is to discriminate the points of the front that are spending a long time in the propagating front, i.e. points that are visited but whose action is not *frozen*, in the sense defined in table 2.1.

Unfortunately, this criterion is very difficult to manage, as shown in figure 8.4. The results are non-predictable, and this is probably because the time spent in the

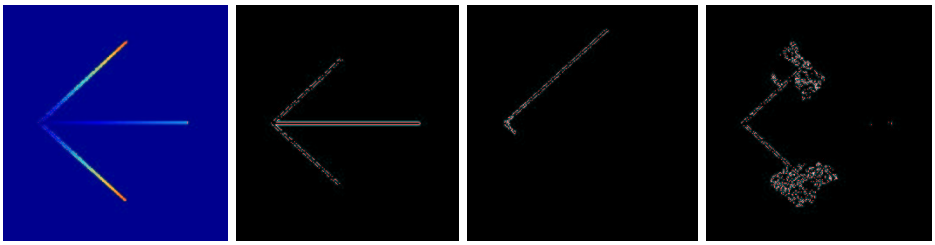

Figure 8.4. Instability of the Time criterion for Freezing: Left image is the action map obtained with a maximum time criterion of 100 iterations; Other images are freezing maps (white pixels) with respectively from left to right 100, 80 and 60 iterations as maximum time spent in the front.

front for a voxel is related to the local cost of the propagation at this voxel, but do not have any relation with the position of the voxel relatively to the object that we

are trying to segment.

Using Distance for Freezing

The distance to the start point is a direct output from the method we already developed for reducing user-intervention in the *Virtual Endoscopy* process in sections 2.2.3 and 3.1. It seems far more "natural" to use the distance to the starting point, or relatively to the most far propagating part of the front, since this notion is completely embedded in the topology of the object we are trying to extract: the section of a tube-shaped objects must be small towards its extent. We must discriminate the points of the front that are near the initializing seed points while other parts of the front are already far. It will prevent from flooding in non-desired area of the data.

We can fix several criterion for the *Freezing* based on the distance. Knowing the current maximum Euclidean path length d_{max} in the front propagation process we can decide that a voxel $\mathbf{v}$ of the propagating front (i.e. *Trial* voxels) should be removed from the front (i.e. set as *Alive* voxel):

- if $\mathcal{D}(\mathbf{v}) < d_{max}/\alpha$, with $\alpha \geq 1$ user-defined; or
- if $\mathcal{D}(\mathbf{v}) < \max(d_{max} - \tilde{d}, 0)$, with $\tilde{d} > 0$ chosen.

The results are now predictable, in the sense that the Euclidean distance to the starting point is a measure which contains information about the geometry of the surface extracted, and in particular its length. This is less related to the local cost of the propagation in each voxel, and more to the position of this voxel in the object. This distance criterion has proven reliability as well in 2D as in 3D, and we worked upon its implementation in the following. A 2D example on the synthetic test is shown in figure 8.5.

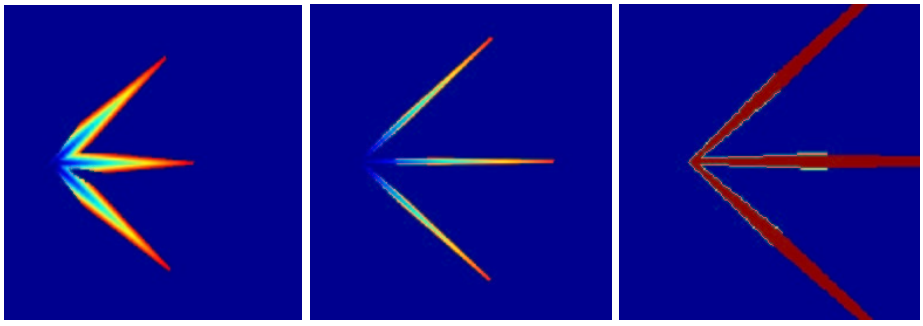

Figure 8.5. Using Distance for Freezing: Left and middle images are action maps with distance criterion of respectively 100 and 50; right image is a zoom on the freezing map for a distance criterion of 50: the pruned points are set in green.

Algorithmic implementation of the *Freezing*

Once the criterion has been chosen, at each time step we insert our visited points both in the classical action related heap, and in another data-structure where the ordering key is the criterion. As for the action, we can use a min-heap data-structure, since the partial ordering provided is sufficient.

At each iteration, we are able to remove all the points whose keys are greater/lower than this criterion, starting from the minimum/maximum element in the tree. It can be implemented easily for the time criterion by recording the iteration at which any point has been inserted in the heap, and to store this time in another min-heap data-structure. Therefore, the element at the top of the heap will still be the point that has spent the longest time without being evolved to the *Alive* set. For the distance criterion, the min-heap key is the computed distance, which means that the element at the top of the heap will still be the point that is the nearer *Trial* point to the starting point.

In the following is detailed an algorithmic implementation of the *Freezing* with the second criterion for the distance information.

Definition

- a starting point $\mathbf{p}_0$, located at the root of the tree structure;
- the usual set of data-structures for front propagation, including an action map $\mathcal{A}$, one min-heap structure $\mathcal{H}_\mathcal{A}$ and a penalty image $\mathcal{P}$ which will drive the front propagation, and which is a function of the position only;
- a distance map $\mathcal{D}$ to compute the Euclidean minimal path length, as explained in section 2.2.3;
- another min-heap data structure $\mathcal{H}_\mathcal{D}$, where the ordering key for any point $\mathbf{p}$ is the value of $\mathcal{D}(\mathbf{p})$, which means that the first element of this heap will be the *Trial* point with smallest distance $\mathcal{D}$;
- several counters d_{max}, $\tilde{d}$, d_{stop}

Initialization

- initialize the classical front propagation method, setting $\mathcal{A}(\mathbf{p}_0) = \mathcal{D}(\mathbf{p}_0) = 0$ and storing the seed point $\mathbf{p}_0$ in both min-heap structures $\mathcal{H}_\mathcal{A}$ and $\mathcal{H}_\mathcal{D}$;
- $d_{max} = 0$
- $\tilde{d}$ and d_{stop} are parameters for tuning the algorithm (user defined).

Loop: at any iteration

- Let $\mathbf{p}_{min}$ be the *Trial* point with the smallest action $\mathcal{A}$;
- proceed according to the classical *Fast-Marching* algorithm, by examining its neighbors, and updating the min-heap $\mathcal{H}_\mathcal{A}$ with the new action values computed;
- take $d_{max} = \max\left(d_{max}, \mathcal{D}(\mathbf{p}_{min})\right)$;
- consider $\mathbf{q}_{min}$, the first element of $\mathcal{H}_\mathcal{D}$, being the *Trial* point with the smallest distance $\mathcal{D}$. While $\mathcal{D}(\mathbf{q}_{min}) < \max\left(d_{max} - \tilde{d}, 0\right)$ do
  - set $\mathcal{D}(\mathbf{q}_{min}) = \mathcal{A}(\mathbf{q}_{min}) = \infty$;
  - set $\mathbf{q}_{min}$ in the *Alive* set, then $\mathbf{q}_{min}$ will not be used for computing the action/distance at its neighbors location.

– delete $\mathbf{q}_{min}$ in both $\mathcal{H}_{\mathcal{D}}$ and $\mathcal{H}_{\mathcal{A}}$;

- if $d_{max} > d_{stop}$, exit the loop.

This heuristic is to discriminate the parts of the front that are propagating slowly, by recording the maximum distance which has been traveled, and compare it to the distance which has been traveled by this parts. If the ratio between those two distances is two important ($>$ given threshold), we "freeze" those parts by setting there speed artificially to zero. It enables to stay inside the object when it is long and thin like tubular structure, as shown in figure 8.5. The domain visited by our algorithm is slightly smaller than the previous one (figure 8.4-right) and this domain shortens with the distance criterion, when we compare left and middle images in figure 8.5. The figure 8.5-right clearly demonstrates than the *Freezing* principle discriminates the points located far from the propagating fronts.

Illustration on the Vascular tree extraction problem

The method explained previously is very useful when it is used for vascular segmentation. Initialization step is therefore performed in a very fast manner by just setting a seed point at the top of the tree hierarchy. Figure 8.6 displays results of this algorithm. The distance threshold is a parameter which is not very sensitive: we generally

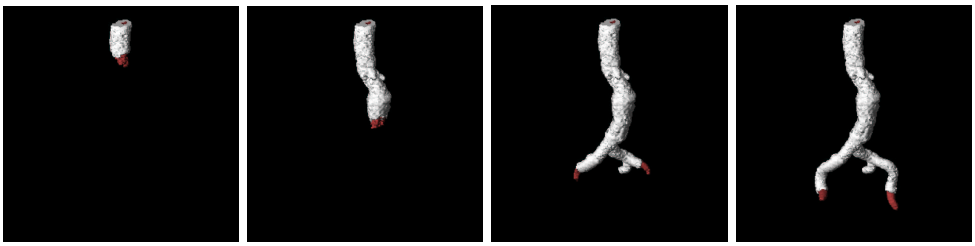

Figure 8.6. Using Distance for Freezing in the Aorta: From left-to-right, images show iterations of the segmentation process; the propagating front is in red, and the frozen voxels are in white.

take a value related to the *a priori* dimensions of the object. This threshold must be more important than the assumed maximum section of the object. It will approximately represent the volume of points bounded by connected envelope of the front voxels that are not frozen.

8.3.2 Suitable stopping criterion

Having designed an adequate criterion for *Freezing* the unwanted parts of the front that could lead to "flooding" of the evolving wave in other parts of the image, we now explain our strategy to stop automatically the propagation.

Previous strategy was to use a maximum Euclidean path length to stop propagation, like for the virtual endoscopy application. In *Virtual Endoscopy* , the user can

set both extremities of the trajectory, if he has an *a priori* knowledge of the anatomical objects. Extraction of tree-like structures cannot use such an assumption: the number of branches in our structure is undefined, only assumption being that the user can fix a point inside the structure, at the beginning of the segmentation process.

The *Freezing* process will provide a criterion which is independent of the number of different branches to recover. If we plot the maximum distance d_{max} of section 8.3.1, as a function of iterations while propagating, we will observe the following profile shown in figure 8.7. We clearly see that this distance increases linearly until a big decrease of the slope appears. It is important to notice that this shock indicates when

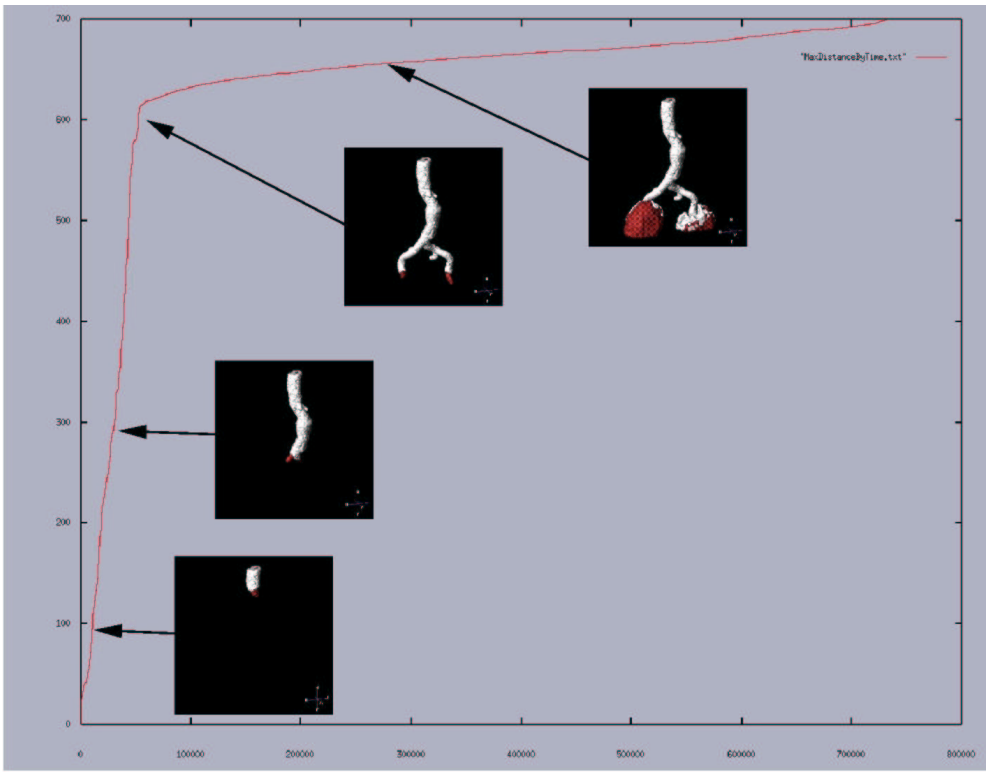

Figure 8.7. Using Distance for Stopping propagation in the Aorta: The images of the propagating fronts of figure 8.6 are super-imposed on the evolution of the maximum distance crossed by the front propagation across iterations; it emphasizes that the decrease in the slope is related to the "flooding" out the aorta.

the front flows out of the object at "heads" of the front. We decide to stop front propagation at this particular time. During the first part of the plot, the function is quasi-linear. The slope is directly related to the section area of the tubular object. By definition of Fast Marching, the number of iterations is equal to the number of voxels that are alive. It means that passing through a certain length in the aorta

implies to visit a number of voxels proportional to the length.

Let us assume that the global section of our aorta is constant in our dataset. This is approximately true in large parts, but becomes a wrong assumption in the very thin parts of the vessels and arteries. But we can assume that the front propagates at the same speed inside the object. Therefore, the number of voxels visited is proportional to the section area. Then the slope collapse can be easily detected using a simple threshold on the slope, depending on the object we want to extract. Even if there are aneurysms in the data set, and even if the mean section of the object increases with the depth, we can assume that we do not want to extract an object which is twice the maximum section. We could then derive a criterion on the maximum section of the object S_{max} which is obviously related to the section area of the object of interest. Recording the first iteration where the front flows out, it gives us the maximum distance where we must stop propagation.

8.4 Extracting the skeletal information

In the following, we assume that we use the *Fast-Marching* and the *Level-Sets* in a collaborative manner, in five steps:

1. the user input is a seed point for region-growing;
2. the *Fast-Marching* using the *Freezing* principle is evolved from this starting point;
3. this evolution is stopped using either the distance, the user intervention, or an automatic criterion;
4. the binary mask defined by the propagation gives the initialization of the region based descriptors k_{in} and k_{out}, as used in section 6.2;
5. the *Level-Sets* model is evolved with equation 4.6 for a small number of iterations.

The process is really similar to the framework detailed in section 5.4. The *Fast-Marching* using the *Freezing* principle will act as a rough initialization step, which will provide the binary image of the voxels visited. This mask will also serve to initialize the different probabilities of the region descriptors defined in section 5.2. First row in figure 8.8 shows the surfaces of several tubular structures extracted with the *Freezing* algorithm. The domain of voxels visited during this first step is used to set correctly the descriptors of the *Level-Sets* model, that converges in a few iterations to the surfaces which are shown in the second row in figure 8.8. Notice that the scheme used here is in equation (7.1), where forces have been included to restore the distance function.

In this chapter, we do not implement dedicated algorithms based on the *Level-Sets* methods. They are used in a very classical manner, to converge to sub-pixel accuracy results, on the basis of as-hoc fast algorithms. However, the level of accuracy that is achieved by the *Level-Sets* cannot be of course outperformed by the initialization

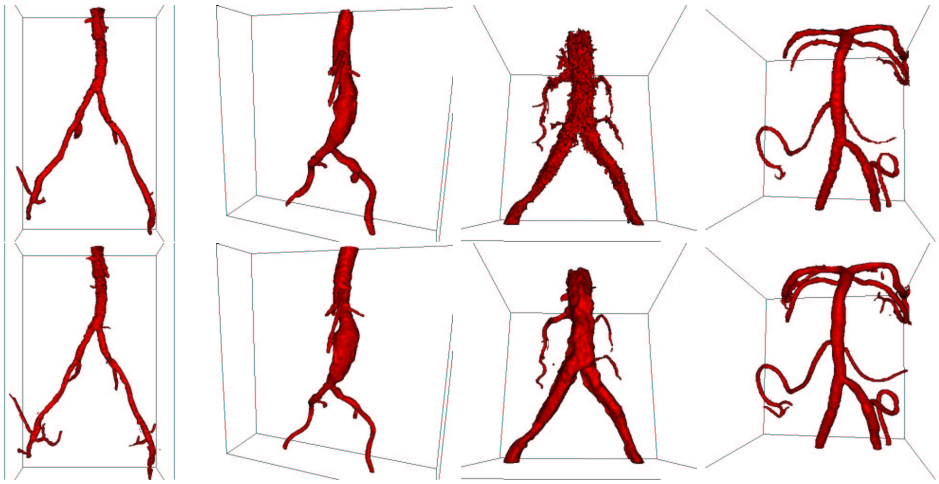

Figure 8.8. Final segmentation of vascular objects: First row shows different vascular objects that have been extracted with the *Freezing* algorithm - except the example shown in last column of the right, where the method used was the competitive fronts algorithm; Second rows is the final result of the segmentation after 40 iterations.

method. The convergence step they achieved cannot be replaced in any way by the *Fast-Marching*.

8.4.1 Combining path and shape extraction

The complete framework for path and surface extraction we have developed will be illustrated in this section by results on a **3D-RA** acquisition of a stenosed vessel, which is shown in figure 8.9.

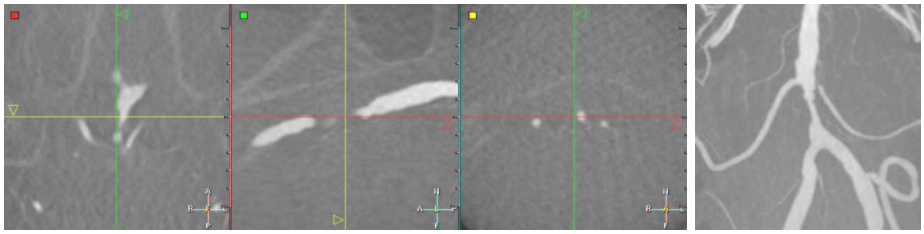

Figure 8.9. 3D-RA dataset of an aortic stenosis: left image shows three orthogonal views of the dataset; right image is a **MIP** view of the same dataset.

We have shown in the part I of the thesis how to extract a trajectory inside a tubular structure. In part II of the thesis, we have combined fast and accurate methods for shape extraction of tube like objects. We now want to combine the

results of both parts, and extend this facility to the detection of tree-like structures, and their corresponding set of multiple trajectories, in order to enhance measures and visualization of pathologies of any tube-shaped object.

We worked upon the extension of the trajectory extraction method, applied for example to virtual endoscopy in chapter 3 to the case of multiple trajectories. For example, the dataset in figure 8.9-left is a structure with branches, which pathology - a stenosis - is clearly visible on the **MIP** view in figure 8.9-right. The complete study of this pathology, with minimum interactivity, would be to extract its surface, and all needed trajectories inside it, in order to give accurate measurements.

Techniques found in the literature

The combination of path and shape representation is a framework already studied as well in Computer Graphics as in Computer Vision. In Computer Graphics, cylindrical shapes description is done by implicit surfaces (in the sense of [13, page 223]) defined by the convolution of a filter kernel with a skeleton. In other words, this *distance surface* is a surface that is defined by distance to some set of skeletal elements, like any curve. But in graphics, the target is to improve visualization and interactivity over the representation of the object. However, it connects to vision because it is often convenient to model a shape as a generalized cylinder as done in [132], for reconstruction of anatomical shapes, as done in [175] by combining the fitting of a generalized cylinder, and its symmetry axis.

In those methods, the central axis constrain the extraction, and models the tubeness of the final object extracted.

Our multiple path extraction method

In our case, the shape is initialized by *Fast-Marching*, thus a path construction method, but we are going to use the solution at convergence of the *Level-Sets* in order compute the final set of trajectories - i.e. the skeletal information of our object. Therefore, shapes controls path extraction. This is exactly the kind of methods that lead to accurate measurements and visualization of the objects:

1. It relies on a sub-pixel shape extraction model; thus the intersection of a cross section plan and the surface is an improved measure of the objects, while cylinders approximate the model.

2. The *Level-Sets* enables any change in topology, and there is no constrain on the initialization of the model, how huge can be the number of branches in the anatomical object.

3. The paths used for quantification are based on this robust surface extraction model, increasing the robustness of the measures.

4. The user input is limited to the setting of the root of the tree hierarchy.

Our method is based on the construction of a connectivity map, by looking at several chosen iterations to the connectivity of the propagating front (i.e. the *Alive* voxels)

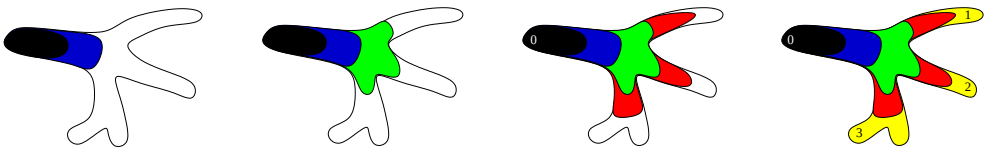

Figure 8.10. Multiple path extraction algorithm: From left to right, these are connectivity tests made on the propagation of a wave inside a segmented object, starting from the voxel designated by label 0, until the whole domain is visited.

and the connectivity of the sets of voxels visited, as shown in figure 8.10. Defining a distance step, each time this step has been accumulated by the front, we label the different sets of visited voxels, and we thus detect when a front separate, at a branch, into several not connected sets.

When the whole domain has been visited, we take for each separate set a representative voxels, which is the most far from the starting point, and we set it as an applicant extremity for back-tracking a trajectory. Notice that the distance step defines the accuracy of the method, since a too important step will lead to misunderstandings: on the right image, only the extremity designated by the label 3 will be eligible for back-tracking, while there are two branches, because the distance step is bigger than both branches.

Multiple Path Extraction Algorithm

The algorithm we devised for multiple path extraction is mostly inspired from works on skeletal extraction from binary, or scattered data. It can be easily compared to morphological processes, but has two advantages: we can choose the *scale* or accuracy of the multiple path extraction, and we can derive this *scale* from anatomical knowledge of the data studied. It is a complete framework in the sense that, the path extraction relies here on a segmentation process which can be as well handled by the *Fast-Marching* or the *Level-Sets* methods. This segmentation step defines a binary mask $\mathcal{M}$ which is one of the main input of our algorithm:

Definition

- a binary mask $\mathcal{M}$ which defines the region of interest in the image;
- a penalty image $\mathcal{P}$ which will drive the front propagation;
- a distance map $\mathcal{D}$, computed with the method described in section 2.2.3, and a distance step d, user-defined parameter that controls the accuracy of the end-point extraction;
- a counter c_d that recalls the iteration number of the loop in our algorithm;
- a label map $\mathcal{L}$ to label each branch detected, $n_{\mathcal{L}}$ a label counter, and an array $\mathcal{E}$ which will recall the hierarchy of the branches detected;
- a starting point p_0, located at the root of the tree hierarchy.

Initialization

- $\mathcal{M}(i,j,k) = 1$ for all voxel in the region of interest, elsewhere $\mathcal{M}(i,j,k) = 0$;

- $\mathcal{L}(i,j,k) = -1$ for all voxel in the image domain, $n_{\mathcal{L}} = 0$, and all elements of $\mathcal{E}[i]$ are set to -1;
- We initialize the usual set of data-structures for front propagation, including an action map $\mathcal{A}$, the distance map $\mathcal{D}$, and a min-heap structure;
- we initialize a classical front propagation method, setting $\mathcal{A}(p_0) = 0$ and storing p_0 in the min-heap structure; item the counter $c_d = d$.

Loop

- we propagate the front with Eikonal equation, computed with penalty $\mathcal{P}$ on the domain defined by the mask $\mathcal{M}$;
- for each *Trial* point p visited in the *Fast-Marching* algorithm, $\mathcal{L}(p)$ is set to the label of its current *Alive* neighbor with minimal action;
- if we visit a voxel p with $\mathcal{D}(p) \geq c_d$:
  1. we consider the set of *Trial* points $\mathcal{T}$, that are all stored in the min-heap data structure, we consider $t_1, \ldots, t_k$ its k subsets of connected components (with 26-connexity in 3D), obtained through a simple connectivity algorithm;
  2. In all subset $t_i, i \in [1, \ldots, k]$
     - considering the old label $l_{i_{old}}$, and the new label $l_{i_{new}}$, we set $n_{\mathcal{L}} = n_{\mathcal{L}} + 1$, $l_{i_{new}} = n_{\mathcal{L}}$, and $\mathcal{E}[l_{i_{new}}] = l_{i_{old}}$;
     - for all the points $p \in t_i$, we set $\mathcal{L}(p) = l_{i_{new}}$;
  3. $c_d = 2 \times c_d$;
  4. we stop if the whole domain defined by $\mathcal{M}$ is visited.

Termination

- we consider all sets $\mathcal{L}_j$, $j \in [1, \ldots, n_{\mathcal{L}}]$ defined by the label map $\mathcal{L}$ with different labels l_j;
- we select the subset of $\mathcal{L}_k$, $k \in [1, \ldots, n_{\mathcal{L}}]$, which have $\mathcal{E}[l_j] \neq -1$ and $\forall n \in [l_j; n_{\mathcal{L}}]$ $\mathcal{E}[n] \neq l_j$;
- $\forall \mathcal{L}_k$ selected, we find the voxel (i,j,k) with maximum distance $\mathcal{D}(i,j,k)$ and set it as end point for back propagation;
- we back-propagate from all final voxel selected and extract a set of multiple trajectories.

Figure 8.11 shows several label maps $\mathcal{L}$ with $c_d = 10$, 30 and 50. c_d is the minimum size of the branches detected, it is the *scale* of the algorithm accuracy. If this scale is chosen small, lots of branches will be detected, but if the scale is increased, the computation time will decrease as well, because it controls the number of connectivity tests which are performed on the *Trial* voxels, during propagation.

illustration on the vascular tree extraction

In figure 8.12, one can observe the complete framework of *Fast-Marching* initialization followed by several iterations, using *Level-Sets* methods, and finally, the extraction of multiple trajectories inside two different datasets. The computations for the paths are restricted to a small number of points, located inside the objects of interest (usually less than 20% of the whole volume, leading to interactive computing times. Those

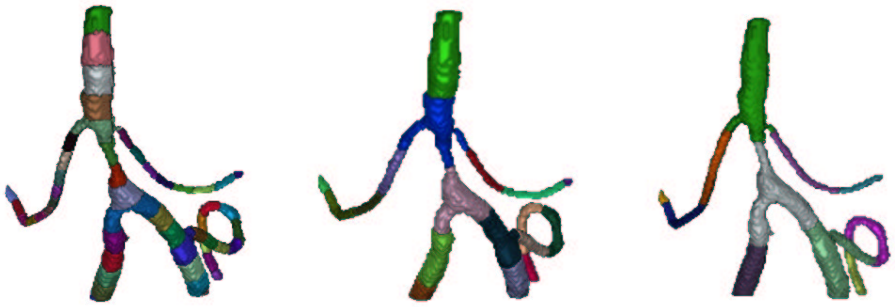

Figure 8.11. Labeling algorithm for multiple path extraction: From left to right, the images show the label map obtained with the multiple path extraction routine applied with path steps 10, 30 and 50 respectively.

paths are already very useful for virtual inspection of pathologies, for example in the aorta (as done in the section 3.1), or measurements along the trajectories extracted, using the techniques detailed in section 7.2. Figure 8.13 shows the result of applying the multiple path extraction algorithm explained previously. This set of paths is the basis of the quantification techniques that can be applied on such a dataset (this aorta presents an Abdominal Aortic Aneurysm).

Originality of this algorithm, towards front propagation techniques applied for multiple path extraction, as in [101], where the set of endpoints is manually drawn in the original image. In our case, all trajectories are extracted automatically.

8.4.2 From Trajectories to Tree Extraction

The trajectories obtained with our algorithm can guide virtual endoscopes. They can also be used for quantification of pathologies, by measuring the variation of the section of the object, across the curvilinear abscissae of the path extracted. But the information of trajectory is not related to the whole branching structure and is just the minimal centered path between two extremities. Therefore, the user is assumed to know the position in the object of this trajectory. And those trajectories are not related to each other, leading to possible misunderstanding in this position. Moreover, this absence of spatial relationship between the paths and the surface disable the use of further developments like automatic labeling of the branches, and accurate localization of pathologies. In order to extract the information which is relevant in order to analyze the surface of the tree-shaped object extracted, we need to extract the underlying skeleton on the basis of our trajectories, as done in figure 8.14. The process of extracting the tree structure from the trajectories is simple: during backtracking of the trajectories, we adjoin those which are close to each other, creating a branching point. The only parameter is the definition of proximity between trajectories.

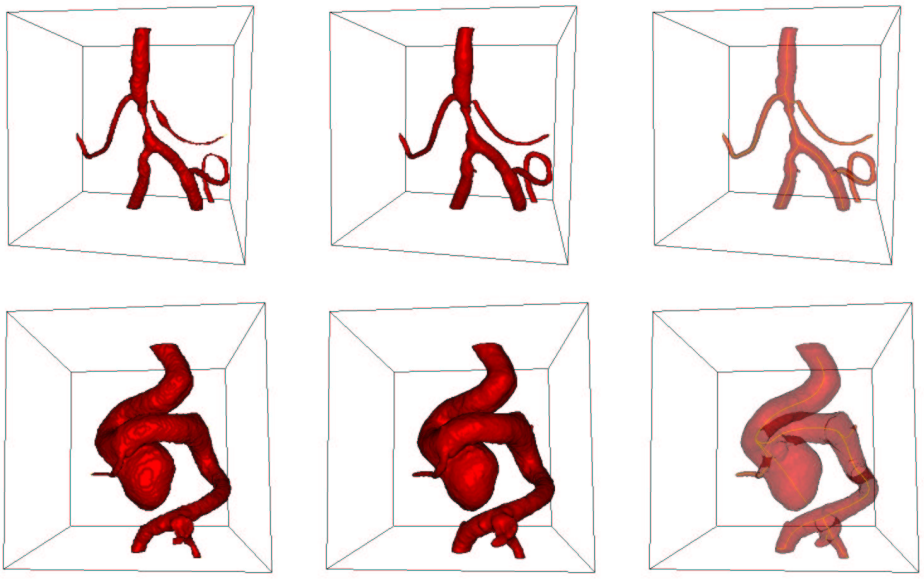

Figure 8.12. Complete method applied to several objects: First row is the framework applied to the stenosed object of figure 8.9 and second row concerns the aneurysm shown in figure 6.4 - Left column is the initialization given by the *Fast-Marching* method; middle column is the surface obtained after a small number of iterations of the *Level-Sets* method; right column shows the multiple trajectories extracted with the labeling algorithm, by transparency.

Algorithmic implementation

As a second process, we can extract a skeleton of our object, from this set of multiple trajectories. The initialization use the same input than the multiple path extraction process, including the final end points extracted.

Definition

- a binary mask $\mathcal{M}$ which defines the region of interest in the image;
- a penalty image $\mathcal{P}$ which will drive the front propagation, usually this penalty map is computed using the centering method described in section 2.3;
- the action map $\mathcal{U}$ computed with this penalty during the initial multiple path extraction;
- the starting point p_0, located at the root of the tree hierarchy;
- the set of end points e_i $i \in [1; N_e]$ where N_e is the number of end points extracted.
- a distance step d which defines the minimum distance between two trajectories (this distance step is chosen bigger than the gradient descent step).

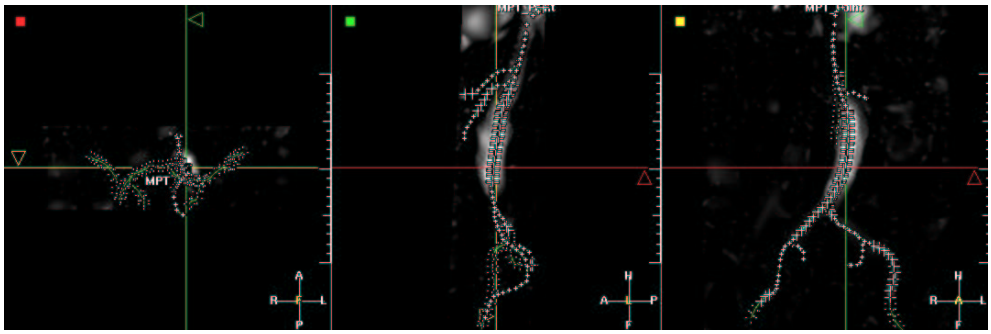

Figure 8.13. Multiple trajectory extraction from only one seed point: This figure represents the projection on three orthogonal views of the complete set of trajectories tracked in the aorta MR dataset which was segmented in figure 8.8 in the second row; the *Freezing* method for initialization with the *Fast-Marching* algorithm has been used to extract centered trajectories.

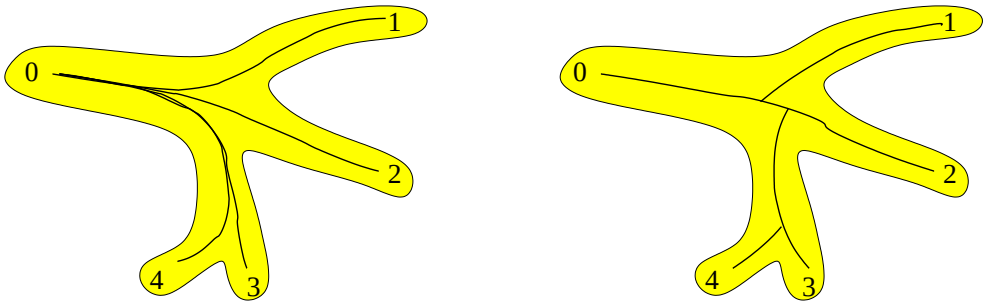

Figure 8.14. From trajectories to tree representation: Left image is a set of trajectories extracted in a segmented object. right image represents the valuable tree structure needed for quantification.

- another different label map $\mathcal{L}$ to label the voxels that are neighbors of a path, which means that the distance between this voxel (i, j, k) and a path extracted is less than d;
- an array $\mathcal{E}$ to recall the branches detected.

Initialization

- $\mathcal{L}(i, j, k) = -1$ for all voxel in the image domain;
- $n_e = N_e$ and $\forall i \in [1; n_e]$, $\mathcal{E}[i] = 0$.

Loop: for $i \in [1; N_e]$

- we back-propagate from e_i, on the action map U using a simple gradient descent method, as described in equation 2.6;
- at every path step, the position of the new path point is defined by $(x, y, z) \in \mathbb{R}^3$
- we consider the vertices of the Cartesian grid that surround $\overrightarrow{x} = (x, y, z)$, the voxels $\overrightarrow{n} = (i, j, k) \in \mathbb{N}^3$ which verify $D_2(\overrightarrow{x}, \overrightarrow{n}) < d$, where D is the Euclidean distance;

- if, for all those vertices $(i,j,k) \in \mathbb{N}^3$, $\mathcal{L}(i,j,k) = -1$, we set $\mathcal{L}(i,j,k) = i$, and continue back-tracking for e_i;
- else, if one of the vertices (i,j,k) verifies $\mathcal{L}(i,j,k) \neq -1$, a branching point is detected, then:
  - recall the label $l = \mathcal{L}(i,j,k)$;
  - $n_e = n_e + 1$, $e_{n_e} = (i,j,k)$;
  - $\mathcal{E}[i] = e_{n_e}$ and $\mathcal{E}[n_e] = 0$;
  - stop back-tracking for e_i;
  - continue back-tracking, this time for e_{n_e}, substituting all $\mathcal{L}(i,j,k) = l$ by $\mathcal{L}(i,j,k) = n_e$, until another branching point or p_0 are found;
- if we reach p_0, then stop back-tracking for e_i.

Termination

- for all end point e_j $j \in [1; n_e]$, we can consider the couples of endpoints $(e_j, e_{\mathcal{E}[j]})$ as extremities of linear parts of the skeleton (with $e_0 = p_0$).
- the multiple paths between couples of points $(e_j, e_{\mathcal{E}[j]})$ $j \in [1; n_e]$ build the skeleton of our object, at scale c_d and distance d.

Figure 8.15 displays the result obtained on the dataset shown in figure 8.9. From

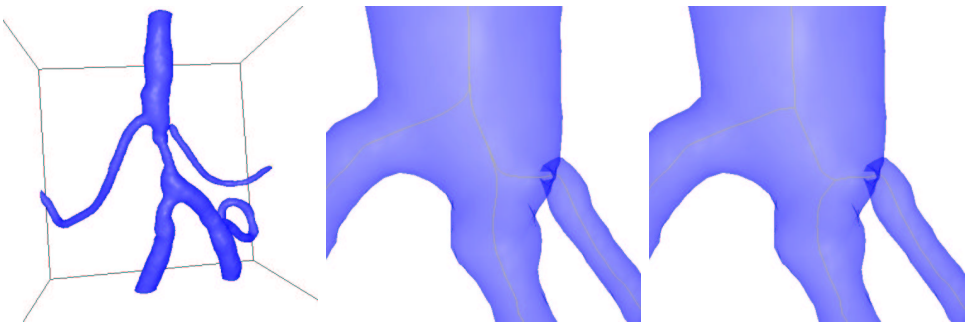

Figure 8.15. Obtaining a tree hierarchy from a set of trajectory: Left image is the segmented object extracted from the dataset shown in figure 8.9; middle image is a zoom on two bifurcations of the object, where the trajectories extracted are displayed; right image is the same point of view on the translucent surface extracted with the tree extracted from the set of paths.

the set of multiple trajectories, branching points are extracted, as shown in figure 8.15-right.

Measurements on the tree

The computational cost of the tree extraction finds its justification in the improvement of the measurements along the new set of trajectories available. Figure 8.16 compares the section measurements with multiple path extraction technique, and tree

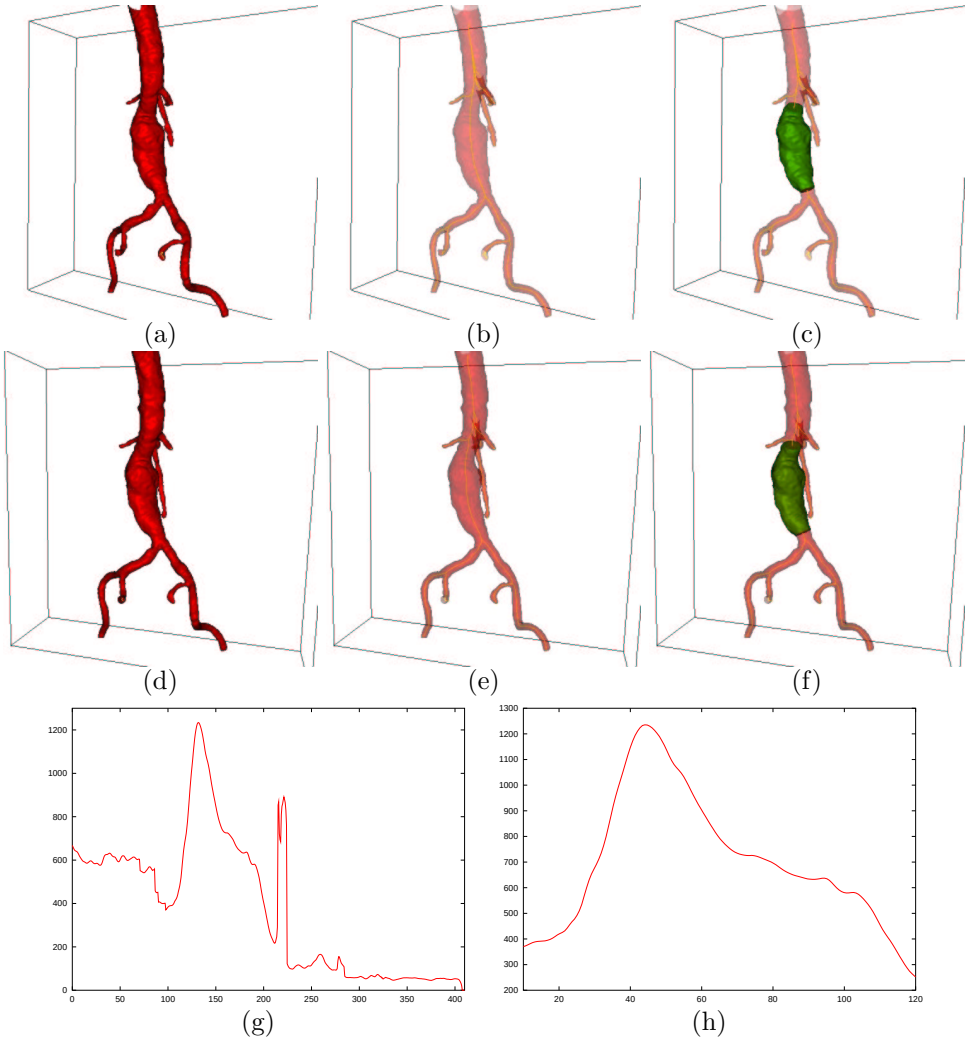

Figure 8.16. Comparing results with the multiple path extraction, and the tree extraction: First row shows images of the segmentation process (a) plus the multiple paths extraction, visible by transparency (b) and the region of interest in green that isolate the aneurysm (c) along one of the trajectories; second row shows the same images (d,e,f) using the tree structure extracted from the same seed point; last rows shows the variation of the section along the paths that are inside the aneurysms, for the complete trajectory (g) and for the branch (h).

extraction technique (dataset shown in figure 3.12). The tree extraction, as shown by transparency on figure 8.16-(e) enables to measure the section along the necessary subset of the object, delimited by the the two branching point (this subset has been colored in green on the figure). If this information is plotted across a trajectory in

the entire object, it is not useful for two reasons

- section information is not valuable at the branching points;
- the position of the part of interest cannot be obtained straightforwardly.

This problem is illustrated in the last row of figure 8.16. The plot of the object section across the curvilinear abscissae of a trajectory is shown in figure 8.16-(g), versus the same plot across a branch of the tree extracted in figure 8.16-(h).